A computational grid devoted to fluid mechanics

Patrick Nivet

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A computational grid devoted to fluid mechanics

Patrick Nivet

N° 0318

February 2006

Thème NUM

ISSN 0249-0803

A computational grid devoted to fluid mechanics

Patrick Nivet

Thème NUM — Systèmes numériques
Projets Smash

Rapport technique n° 0318 — February 2006 — 28 pages

Abstract: This document describes the establishment of a computational grid devoted to fluid mechanics, using three clusters of PC localized respectively at INRIA Sophia-Antipolis, CEMEF Sophia-Antipolis and IUSTI Marseille. We will expose the encountered problems and the choices retained according to the computational ressources. In a first part 2, we briefly describe the MECAGRID project that aims to set-up a computational grid devoted to massivelly parallel computations in fluid dynamics. The starting points of the project are several clusters localized in the PACA area already used by several teams working in CFD. One of the main requirements of this project has been to be able to launch on this grid several codes using MPI with no modification of these codes. Then, we present an analysis of possible solutions for establishing this grid in chapter 3 and a description of the selected system. The encountered problems are detailed throughout this chapter. We finally show some preliminary results in chapter 4 and some examples of computations in chapter 5.

Key-words: computational grid, GLOBUS, cluster, MPI, CFD, parallel computations, message passing, Fluid Dynamics

Unité de recherche INRIA Sophia Antipolis

Mise en place d'une grille de calcul régionale dédiée à la mécanique des fluides

Résumé : Ce document décrit la mise en place d'une grille de calcul régionale dédiée à la mécanique des fluides, comprenant le cluster de l'INRIA Sophia, du CEMEF et de l'IUSTI. Nous y verrons quels ont été les problèmes rencontrés ainsi que les choix retenus en fonction des moyens de calcul dont nous disposons.Tout d'abord nous allons brièvement décrire le projet MECAGRID dont le but est de mettre en place une grille de calcul dédié au calcul parallèle en mécanique de fluides dans le chapitre 2. A l'origine de ce projet, il y a plusieurs grappes de calcul situées en région PACA et déja utilisées par des équipes travaillant en mécanique des fluides. L'un des principaux buts de ce projet était de pouvoir lancer plusieurs codes de calcul utilisant MPI sans modification de ces codes. Par la suite, nous présentons une analyse des solutions possibles pour établir cette grille dans le chapitre 3 ainsi qu'une description du système retenu. Les problèmes rencontrés sont détaillés à travers ce chapitre. Finalement, nous présentons quelques résultats préliminaires dans le chapitre 4 et des exemples de calculs dans le chapitre 5.

Mots-clés : grille de calcul, GLOBUS, grappe, MPI, éléments finis, parallélisme, passage de message, mécanique des fluides

1 Introduction

Network based computing has recently appeared as a new way to offer end-users a dramatic increase in available processing power. Beside deceptively parallel applications like SETI@home that have made popular the concept of grid computing, the rapid pace of the improvment of network connectivity and speed makes it possible to consider communication-intensive parallel jobs (now currently run on dedicated supercomputers) as suitable for execution on "computational grids" consisting of geographically distributed compute-and-data storage ressources connected by various communication links. The success of these so-called computational grids relies on several factors : the rapid increase of the CPU power and network connectivity improvment, the growing availability of middleware dedicated at building grids of processors and storage ressources, the design or re-design of grid enabled parallel algorithms and last, but not least, the participation and involvment of the end-user communities.

Today, a lot of middlewares dedicated to grid computing are available but the set-up of a computational grid requires a careful examination of the different possibilities and of their limitations according to the needs of the end-users. A middleware is a general term for any software that serves to "glue together" two separate and usually already existing programs. A middleware dedicated to grid computing is the glue used to make parallelism on a set of clusters or shared memory multiprocessor machines. The most widely known middleware is certainly the Globus Toolkit developped at Argonne National Laboratory but many others exist as Legion, Unicore or ORB implementations. This report contains a collection of cogitations issued from the installation phase and a description of the installed system dedicated to the execution of Computational Fluid dynamics (CFD) parallel applications using the MPI message passing library.

This paper is composed of three parts. First, we briefly describe the MECAGRID project in chapter 2. Then, we speak about its objectives (the setup of a computational grid devoted to CFD) and what we start from (several clusters in PACA area already used by local scientists and several codes using MPI must be launched on this grid). Then, in chapter 3 this report presents an analysis of possible solutions for building this grid and a description of the selected system. The encountered problems are detailed throughout this chapter. We finally show some preliminary results in chapter 4 and examples of computation in chapter 5 with AERO3D and STOKES.

2 The Mecagrid project

Mecagrid is a project of the french ministry of research throught the ACI-GRID program. It is a joint project between INRIA-Sophia, CEMEF of the ENSMP (Ecole des Mines de Paris) localized in Sophia-Antipolis and IUSTI of the University of Provence localized in Marseille. We present briefly the objectives of the project, the requirements of the end-users, the existing computational ressources and we review the technical problems that we have to solve. This reflexion is useful because it shows us the gap between the initial state of the clusters and softwares, and what we have to obtain, revealing technical problems.

2.1 Objectives

The aim of the project is to build a computational grid devoted to fluid mechanics, using a set of three clusters interconnected by a wide area network (internet). The total number of processors is 166 which makes possible the computation of larger problems than on a single cluster. Moreover, the availability of this number of processors can allow a user to use distant nodes in the event that the number of processors needed to perform the computation is not availabel locally.

To reach the objectives of the Mecagrid project, two different approaches can be choosen:

- TRANSPARENT SOLUTION: Build a computational grid that simulates one virtual parallel computer as a single cluster: however, for this, we have to be careful of network and node heterogeneity.

- COMPONENT ORIENTED ARCHITECTURE: Take advantage of the specificities of the grid by using Object Request Broker technologies. In other words, re-write the software in an ORB way.

2.2 Pre-existing softwares

At the beginning of this project, several application codes were existing as AERO3D from INRIA or STOKES from CEMEF. These codes, written in FORTRAN and C++, have a common characteristic: they use MPICH [GL96] library and probably it will be the common point of most application codes thus it is an important information. Another code, AEDIPH, is being written for two-phase flows problems. Initially, these codes were designed to run on homogeneous clusters so, in a grid deployement perspective, it is planned to integrate load-balancing capabilities.

As a significant characteristic of the application codes that that have to be run on the grid, MPI [WD96] standard and MPICH implementation of MPI will be detailed:

- MPI [WD96] is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users which specifications are available as a "MPI standard". MPI was designed for high performance on both massively parallel machines and on workstation clusters. It is based on distributed memory model and explicit control of the parallelism.

  The advantages of MPI are:

  - no data placement problems
  - implicit synchronisation with subroutine calls
  - portability to distributed and shared memory machines

  however, this approach has also some drawbacks:

  - no so easy development and debugging
  - high level communications (performance)
  - large code granularity often required to minimise latency
  - global operations can be expensive

- MPICH [GL96] is a portable implementation of MPI, the Standard for message-passing libraries.
  The software architecture of MPICH pursues two main goals, portability and high performance. For portability, the designers of MPICH wished to maximize the amount of code that can be shared. In this way, many of the complex communication operations can be written in terms of lower-level ones. For performances, they wished to provide a structure whereby MPICH could be ported to a new platform quickly, and tuned easily.
  To achieve these goals, they designed MPICH around a central mechanism called the Abstract Device Interface (ADI). All MPI functions are implemented in terms of the macros and functions that constitute the ADI.
  Thus, on one side of ADI, we have the portable part of the library, optimized for performance, and on the other side, we have hardware-dependant implementation of the device. MPICH contains many implementations of the ADI, which provide portability and hardware optimizations.

2.3 Computational and network resources

Computational nodes available for this project are divided into 3 sets, 3 clusters located at INRIA-Sophia, CEMEF and IUSTI. Each site is connected to the others two by classical Internet network. Caracteristics of clusters are detailed in table 1. The cluster of INRIA has 19 biprocessors PIII at 933 Mhz (pf nodes) and 16 biprocessors XEON at 2Ghz. The

cluster of CEMEF has 32 biprocessors PIII at 1Ghz. The cluster of IUSTI has 32 monoprocessors PIV at 2 Ghz.To summarize, all nodes are Intel processors under Linux OS, but kernel versions, batch schedulers, processor speeds and network characteristics are different.

CEMEF at Sophia-Antipolis	
Nodes	: 32 bipro Pentium III (1GHz) with private IP
OS	: Linux Red Hat 7.1
Network	: 2 networks : Myrinet 2 Gb/s + Fast Ethernet 100 Mb/s full-duplex
Parallel libraries	: MPICH 1.2.8 Argonne Myrinet and MPICH 1.2.5 fast ethernet
Batch scheduler	: OPEN PBS
Compilers	: Portland, f90 , f77, C, C++ and GNU f77, C, C++
INRIA at Sophia-Antipolis	
Nodes(pf)	: 19 bipro Pentium III (933 Mhz) with public IP
Nodes(nina)	: 16 bipro Xeon (2 Ghz) with public IP
OS	: Linux Red Hat 2.2
Network(pf)	: Fast-Ethernet Network 100 Mb/s full-duplex
Network(nina)	: Gigabit-Ethernet Network 1 Gbps full-duplex
Parallel libraries	: PVM 3.4.2, MPICH 1.2.5 and LAM/MPI 6.5.4
Batch scheduler	: LSF
Compilers	: Portland, f90, f77, C, C++ and GNU f77, C, C++
IUSTI at Marseille	
Nodes	: 32 Pentium IV monopro (2 GHz) with private IP
OS	: Linux Red Hat 7.2 kernel 2.4.17
Network	: Fast-Ethernet Network 100 Mb/s full-duplex.
Parallel libraries	: MPICH 1.2.5
Batch scheduler	: OpenPBS
Compilers	: INTEL f77, f90, C, C++ and GNU f77, C, C++

Table 1: Table of characteristics for each cluster

Due to this particular architecture, some points need a particular care. They are summarized below:

- Using classical Internet network (i.e. not specialized or secure line) to interconnect clusters is a security hole,
  - during authentification, it implies the use of crypted certificates or strong passwords.
  - during computations because packets will circulate through machines and routers whose security integrity is unknown we have to use cryptographic protections.

- Two of the three clusters (CEMEF and IUSTI) are built on LAN (Local Area Network). It implies that nodes are identified by a private IP address, but communication between two nodes from different sites, as required by MPI, becomes impossible. Indeed, a computer in a LAN is not known and not routable through the Internet.
- This set of clusters is a merge of 3 distinct authorities,and thus each user of the grid will have several identities. In our example, 3 clusters implies 3 authorities and 3 identities per user. During submission and computation, the system must be able to associate the right local user id to a request from another site (same physical person but different logical id). The use of X.509 [TC88] certificates seems to be a good solution.
- The last thing to note is that we have different batch schedulers (OpenPBS [ope03] for IUSTI and CEMEF, LSF [lsf01]for INRIA). These batch schedulers are unable to cooperate. Two workarounds are possible: on one hand, as will be described in 3.1.1, we can make a specific request on each site and, on the other hand, we can use a tool (sort of parser) able to discuss with each batch scheduler. The use and interest of this type of "meta-scheduler" will be discussed in section 3.1.4.

3 Establishment of the grid

3.1 Possible solutions

Now that we described the problem, holdings and outcomes, some possible solutions will be discussed below. The first try consists in a "hand-made" solution using SSH for security purpose. Then our interest will be focused on various MPICH overlayers able to modify communication behaviour of processes to deal with the specificities of our grid. Then we will speak about ORB technology and will finish with middleware solutions.

3.1.1 "hand-made" solution with SSH

The first try was to see if a very simple solution using only MPICH and SSH [Ylo96] for security is possible.
Using SSH with MPICH is something very usual and simple. We just have to be aware of the keys generation protocol (we must generate couples of private/public keys and put public keys in right places) and agent mechanism (it allows an authentification proxy to answer to multiple anthentification challenges automatically).
But it becomes more complicated during submission. On each site, we have an autonomous and different batch schedulers (LSF, OpenPBS,...) so we are unable to make them work together. Before a submission, we must be connected to each frontend, submit an interactive request for nodes, get back FQDN name of furnished nodes and write a procgroup file for MPICH. After that, we are able to launch all MPI processes on clusters.
Another problem is that it does not resolve the fact that MPICH devices need to open direct connexions from node to node by using tcp connexion between two public IP adresses.

The attempt to elaborate an "hand-made" system shows us a lot of problems and what should be characteristics of our solution.

3.1.2 Wrappers as PACX-MPI, MADELEINE III

Wrappers solve problems about multiple network devices on a grid and optimize the choice and use of a collection of devices. Two examples of wrappers are PACX-MPI and MADELEINE
PACX-MPI [pac03] is an MPI-compliant library used for heterogeneous grid environnement and is developped in the High Performance Computing Center of Stuttgart. This library has an interesting feature: daemons, running on each frontend, are used to merge all connections between two clusters in one connection. The first motivation was to reduce the number of connections between SMP but this feature makes possible MPI programs to communicate from or towards nodes with private IP which is very interesting for our configuration. But this solution is also a disadvantage because it requires to launch 2 daemons per submission, as user, on each frontend which does not constitute a regular use of the frontend and batch scheduler.
MADELEINE III [AM], is used to optimise communications between several clusters of workstations. In spite of using standard protocol stack (for example TCP with MPICH-G2

[Kar] or PACX-MPI), MADELEINE is designed to exploit, to its maximum, the characteristics of specialized networks between clusters as like as Myrinet or SCI. Today, no assumption is possible on the nature of inter-cluster links, thus we are unable to use MADELEINE specificity.

These wrappers, solve the problem of private IP adresses because it allow us to modify the inter-cluster connexions. The main drawback of all these wrappers is the same as "hand-made" solution: we have to deal with one different authority and batch scheduler on each site. It implies a manual launch and coordination of processes.

3.1.3 Using an ORB

An object request broker (ORB) is a middleware technology that manages communication and data exchange between objects. In this approach, distinct features are encapsulated in objects localized somewhere on the grid. The broker has in charge to connect requests to services. This solution is very natural for code coupling but it is not totally suitable for massively parralell applications. Moreover, in the Mecagrid project, this solution should have required an important re-engineering of each software and was not found practical due to the large number of codes that the end-users want to run on the grid.

3.1.4 Using a middleware

Concerning middlewares, a lot of solutions are possible. Here we will discuss about Globus which is the most popular, Unicore, Legion, Sun Grid Engine and Condor.

GLOBUS [FK97] is a middleware for grid computing designed on three elements necessary for computing in a Grid environment. The first is Resource Management and it involves allocating resources provided by a Grid. The second is Information Services and it provides information about Grid resources. The third is Data Management and it involves accessing and managing data in a Grid environment.

Resource Management involves the allocation and management of Grid resources. For example, it contains facilities to make different batch schedulers cooperate as LSF or OpenPBS. Information Services provides information about Grid resources. Data Management involves the ability to access and manage data in a Grid environment. For example, it includes components such as GridFTP, which is used to move files between Grid-enabled storage systems.

Globus contains a lot of tools useful for grid computing. Moreover, a specific device for MPICH was implemented and could explain it's popularity. We just have to notice that it does not solve our problem of communication between nodes using private IP adresses.

Unicore[ES01] was developped at Offenbach in Germany by UNICORE Forum association. This project aims the following main goal: developing a middleware for a seamless secure, and intuitive access to distributed resources for german university and research laboratory. The set of tools provided by Unicore is the most complete as it goes from low-level routines to end-user applications such as GUI, plug-ins for scientific codes. But not all low-level tools needed by grid computing are implemented. It seems to be an interesting alternative to Globus and should be studied more precisely.

Legion[CKKG99] falls into the same category as Globus. The main goal of Legion is to generate the illusion of a single, distributed computer. The LEgion Project was, initially, a project of the University of Virginia. Unfortunately, this project became a commercial product named AVAKI from AVAKI Corporation, located in Massachusetts.

Sun Grid Engine[gri03], Condor[TTL02] are not, actually, grid middleware. They can be seen as metaschedulers and can do the same job as LSF or PBS in a distributed environnement.

3.2 Description of our solution

We decided to build a computational grid that simulates one virtual parallel computer, more precisely one virtual cluster. The biggest advantage is we just have to put all our efforts on one thing: the elaboration of a virtual cluster to minimize software adaptation. In spite of spending a lot of time on modifications, re-engineering, for *each* software we want to use on this grid, we spend a finite amount of time once for *all* MPICH based software. Potentially, all FORTRAN, C, C++ codes based on MPICH library are executable on the grid (keep in mind the heterogeneity of this grid). This has been found to be a decisive advantage over ORB Technology. The chosen solution is constituted by a middleware (Globus) over a VPN [Con].

VPN To solve the problem of the clusters in private classes of IP, the communications will be routed through a virtual network (VPN) installed between all frontends.
Some of the clusters constituting this grid are built on LAN. Thus communication between two arbitrary nodes from different sites, as required by MPI becomes a problem. Indeed, a computer in a LAN is not known and not routable through Internet; TCP packets cannot circulate on the global network if one of the addresses (source or destination) is private. Thus, we have to find a way to enable the communication between nodes. Three solutions are possible:

- assigning public IP addresses for each node

- using a middleware for the grid which integrates a proxy running on each frontend.
- using a workaround applied to the network in order to enable the routing between all nodes.

The first one has not been found possible by system administrators since it generates more potential security holes and requires more attention. The second one reduces the possibilities for the choice of the middleware. The third one can be implemented with techniques of port forwarding or VPN (Virtual Private Network). So which one between port forwarding or VPN is better? The solution consisting in using the port forwarding is very heavy to implement and seems to be incompatible with the use of libraries such as MPI which manages connections between the nodes in an autonomous way. On the opposite, the installation of a VPN is fully compatible with the use of MPI because it provides a totally transparent routing for message-passing libraries. It gives us a maximum of possibilities to change and experiment other middlewares or ORB solutions, and avoids the stacking of MPICH layers. This is thus the chosen solution.

The definition of a VPN, given by the Virtual Private Network Consortium (VPNC) is

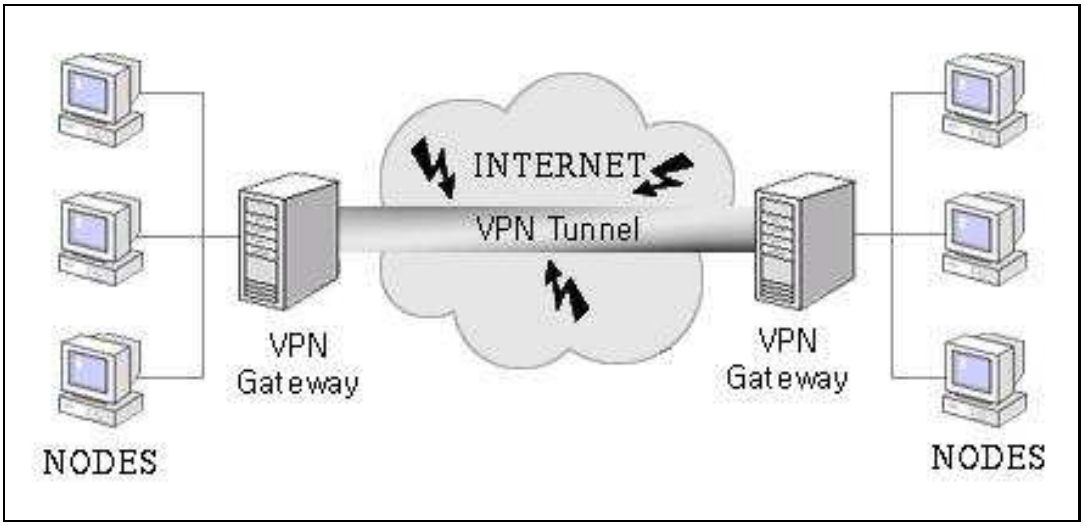

Figure 1: A VPN connexion between two frontends.

the following. It is a private data network that makes use of the public telecommunication infrastructure, maintaining privacy through the use of a tunneling protocol and security procedures. See Figure 1.

During the installation of the VPN, no new private IP address has been given to each node but we simply established routes through the grid and tunnels between frontends in

order to provide connectivity between each couple of nodes. This implies that any private IP address has to be unique amongst the whole nodes. This can be guaranteed by the use of different subnet numbers on each site (here, we use the class B private subnet). More

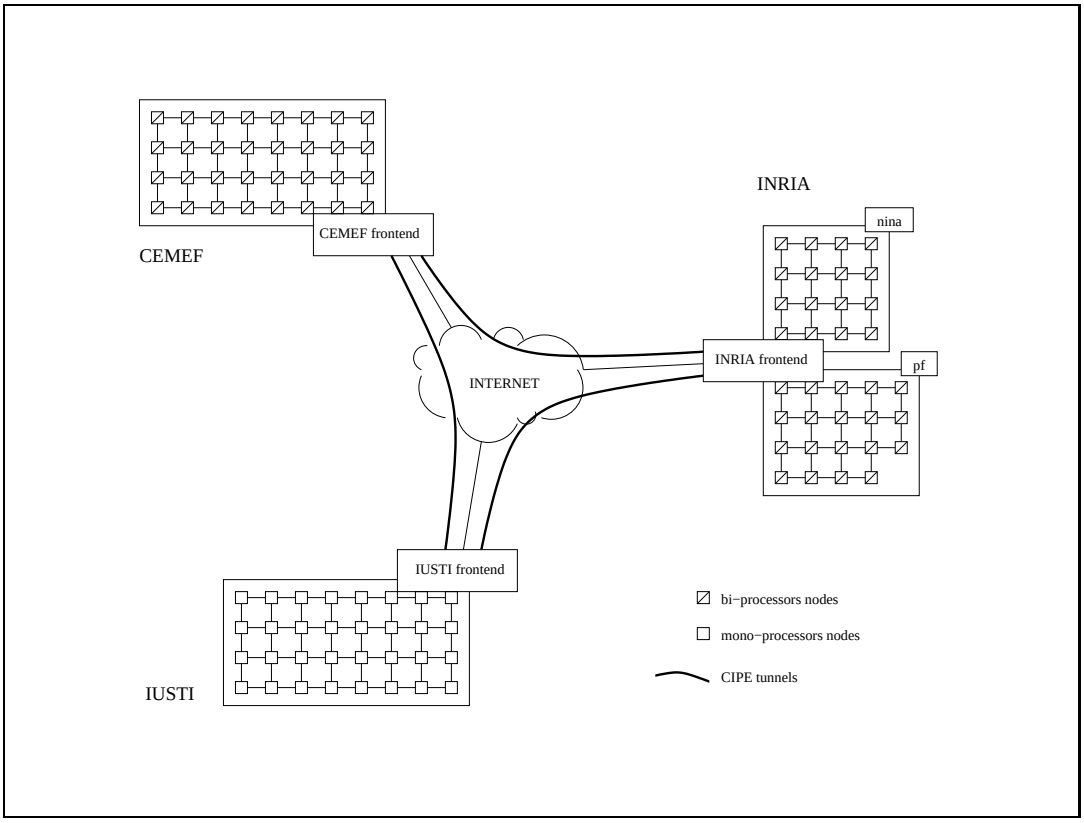

Figure 2: Interconnection of the frontends: a tunnel for each couple of frontends is used to carry TCP packets from or to private IP address.

specifically, as we can see in Figure 2, we have three frontends with public IP addresses. The INRIA cluster consists of nodes with public IP addresses but the CEMEF and IUSTI clusters are made of nodes with private IP addresses. It means that each couple of frontends needs to be connected by a tunnel, in which crypted and encapsulated packets are transmitted. For this, we use CIPE [Tit03], a very simple tunneling software where IP packets are transmitted in encrypted UDP packets on user-defined ports. Furthermore, a compression on the payload of our packets can be applied to improve the flow rate of the connection.

The VPN needs to be completed by the addition of routes. A node must be able to send a packet to another node of our grid and see each frontend as a default gateway for external

addresses (i.e. not on the same LAN). For instance, a packet from an INRIA node to a CEMEF node is sent to the INRIA frontend. On the frontend, special routes are set up to send the packet in the appropriate tunnel. In our example, the packet is sent to the virtual interface of the INRIA/CEMEF tunnel. Then the packet is received by CEMEF frontend and distributed on the LAN.

The VPN provides a set of 166 nodes locally connected in a virtual way but it does not resolve other problems such as scheduling, authentification, unicity of storage space. This second part is taken care by the Globus middleware and is described in section 3.1.4.

The Globus middleware Over this VPN, Globus middleware (see 3.1.4) was installed and works as if all nodes were on WAN. We chose to use Globus. This middleware enables us to make some preliminary tests with a cluster located at the INSA Lyon and already using Globus. The installation of Globus reveal us some problems that were fixed during the first tests. In fact, a big part of our specific problems have been solved during the installation of the VPN. A specificity of our installation is the managing of packets routing. As we saw with MPICH, connexions between nodes (private IP adresses) must use CIPE tunnels but all connexions between frontends are performed outside this tunnel and require to adjust filtering rules.

Timing The installation of the VPN and Globus middleware started in June 2003, after a test and cogitation phase from February to May 2003. The first run on the complete grid has been performed in the middle of January 2004. Since this date we work to improve performances, robustess and usage of our grid.

4 Measure of performances of the grid architecture

In order to evaluate the performance of the grid, a measure tool was implemented as part of the DEA of Rodolphe Lanrivain [Lan03]. Below, we report on the results obtained by the use of this tool during performance of 24 hours where the periodicity of the measures was of 30 minutes. These tests show us the level of heterogeneity and possible variations along time for network and processor speeds.

Acknowledgements This results were provided by Olivier Basset.

4.1 Processor performances

Measured processor speeds in table 2 match with processor characteristics but Figure 3 shows us significant variations for INRIA-nina nodes. A mean standard deviation of 3% for these processors was observed while it is only of 0.2% for the other nodes. This particularity was not observed during other tests. It probably has been generated by a problem of scheduling (we noticed sometimes processes continuing to run even if the corresponding job is finished for the batch scheduler. This biaises measures). We chose to keep this "bad" test to highlight the interest of an adaptative load-balancing along time in real computations.

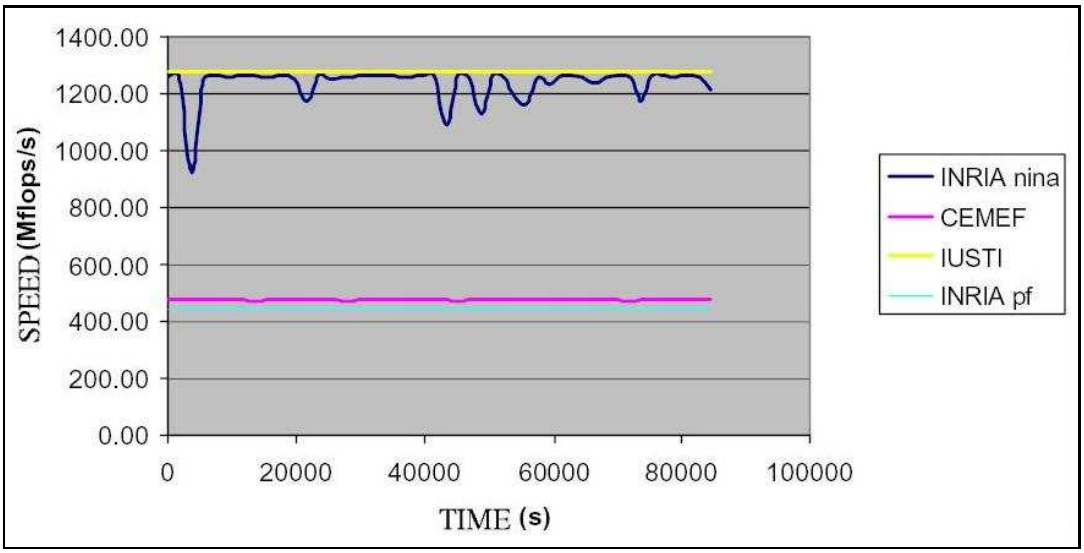

Figure 3: Processor speeds along time.

Processors	INRIA nina	INRIA pf	CEMEF	IUSTI
Average processor speeds (Mflops)	1257.01	444.51	475.95	1275.24
CPU frequencies	2 Ghz	993 Mhz	1 Ghz	2 Ghz

Table 2: Measured processor speeds (average) and CPU frequencies

4.2 Network performances

External network performances Variations of speed on external links are important along time and we can see on Figure 4, that the mean standard deviation reaches 15%. The fact that these links are not dedicated to the grid (other users are present on each site and use some bandwidth) explains these variations. Again, it shows us the necessity to adapt load-balancing during the job.

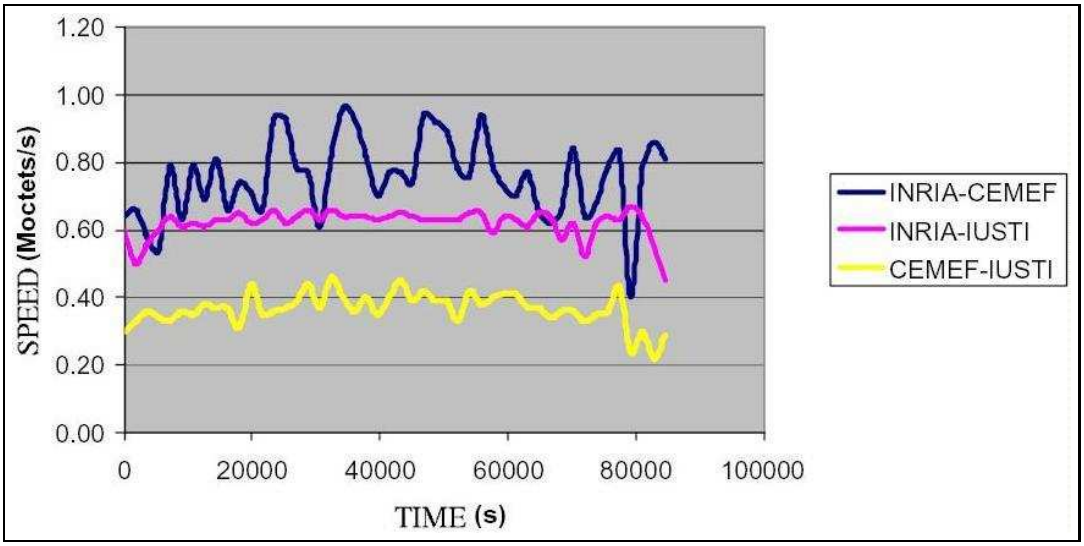

Figure 4: External network speeds along time.

External links	INRIA/CEMEF	INRIA/IUSTI	CEMEF/IUSTI
Average network speeds (Mo/s)	0.86	0.6	0.44

Table 3: External network speeds (average)

Internal network performances: If we look at the speed of internal networks for INRIA-pf, CEMEF and IUSTI, values match with theoretical specifications.We obtain a mean speed superior to 10 Mo/s on 100 Mbit/s networks (see table 4). Concerning INRIA-nina nodes, performances compared to theoretical characteristics are not so good. We have a mean speed of 60.76 Mo/s on a 1 Gbit/s network and variations (see Figure 5) are very important (about 40 Mo/s to 80 Mo/s). It gives a mean standard deviation of 28%, to compare to 0.5% for the others. It confirms that a problem of processor availability during computation is possible.

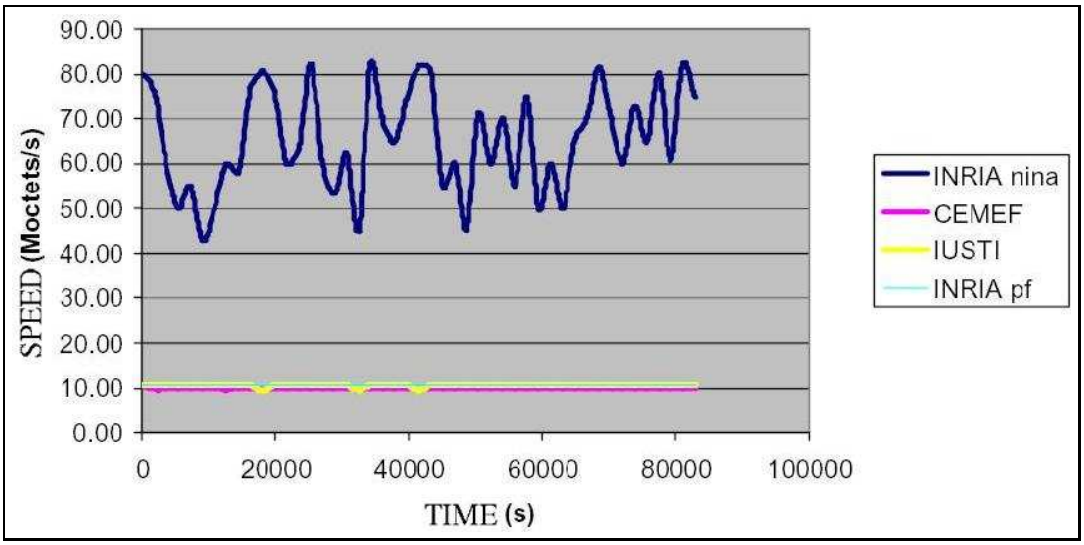

Figure 5: Internal network speeds along time.

Internal links	nina/nina	pf/pf	nina/pf	CEMEF	IUSTI
Average network speed (Mo/s)	60.76	10.29	10.64	10.02	10.33

Table 4: Internal network speeds (mean)

Internal network topology The INRIA's cluster mixes Fast-Ethernet and Gigabit-Ethernet interconnections technologies. "pf" nodes are connected at 100 Mbps and "nina" machines are connected at 1 Gbps. Frontend machine is connected with "nina" nodes. These two network switches are interconnected at 2 Gbps using two 1 Gbps aggregated links. Access to the outside world is done via a 1 Gbps link to Internet (RRTHD or RENATER). RRTHD link provides a speed connexion up to 100 Mb/s and RENATER link provides a

speed connexion up to 2.5 Gb/s.

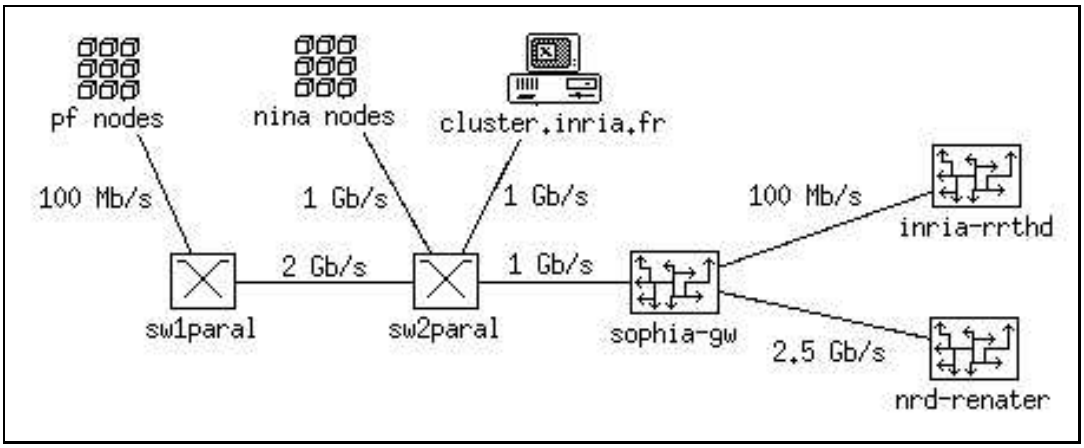

Figure 6: Internal network at INRIA.

The CEMEF's cluster uses Fast-Ethernet interconnection between nodes and frontend. Access to the outside world is done via a 1 Gb/s link to RRTHD router. The RRTHD link provides a speed connexion up to 20 Mb/s and is shared with other ENSMP's laboratories.

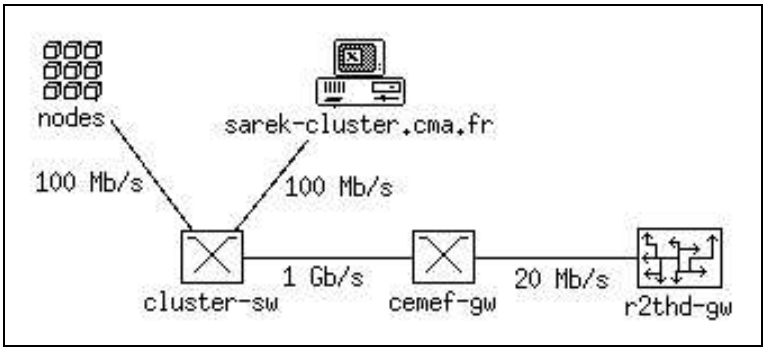

Figure 7: Internal network at CEMEF.

Currently, the internal network topology concerning IUSTI's cluster is unknown.

Problems to resolve Network capabilities are very important for grid computing and we should work on the quality of connections between the 3 sites by reviewing all the devices which establish these connections and improving the worst ones. Network performances between the 3 sites are very poor, see chapter 4. Several explainations can be provided: an internal device on LAN, slows down communications, the connexion to WAN is too slow or this connexion is shared between several laboratories.

After investigations, problems can be summarize as follows:

- INRIA: None, we use a complete 100Mb/s connexion to WAN. Preformances are good.
- CEMEF: We use a low connexion to WAN (20Mb/s), shared between several laboratories.
- IUSTI: We have an Internal problem on LAN between frontend and WAN connexion that slows down communications and WAN connexion (100Mb/s) is shared between several laboratories.

For CEMEF, the upgrade of the connexion (currently 20 Mb/s) consists in the increase of speed connexion to RRTHD or the rent of a specialized line between CEMEF and INRIA. First, an upgrade of RRTHD connexion was considered: the cost (June 2004) of connexion to RRTHD at 1 Gb/s is 15319 EUR per month (6250 EUR per month for CEMEF and 9000 EUR per month for CG06 if accepted). Financially, this solution is impossible. For the second solution (June 2004), a specialized line (Completel) between CEMEF and INRIA costs 1 500 EUR per month for 100 Mb/s and 3 000 EUR per month for 1 Gb/s. It is yet too expensive for CEMEF.

The problem of IUSTI was solved by analysing local area network to find the deficient device. For information, we used a 100 Mb/s connexion but maximum speed measured is 4.8 Mb/s. Since 12/13/2004, the problem is solved and the maximum speed is 76.8 Mb/s. New computations should be faster between INRIA and IUSTI.

5 Example application

AERO3D The AERO3D code is a Fortran code which solves the 3D compressible Euler or Navier-Stokes equations on unstructured tetraedral meshes using a mixed Finite Element/Finite Volume method. Features of the code include turbulence modelling (by the k-ϵ model), mesh deformation, explicit and implicit timestepping. The code is second-order accurate and is fully parallelised using the MPI message passing library. Its range of applicability goes from subsonic flows to hypersonic ones. This code is mainly used for aerodynamical computations for complex geometries in internal or external flows.

The following example is a wing mesh of 22.000 vertices. The final solution was computed in 128 steps on a set of 8 processors.

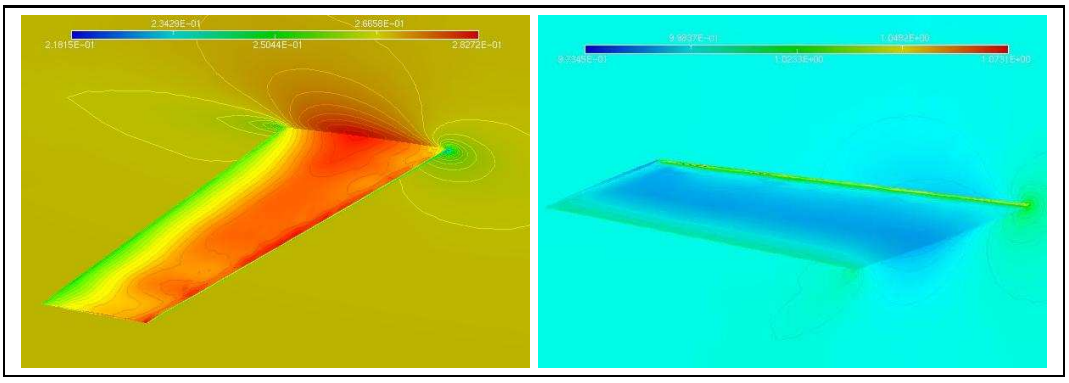

Figure 8: Mach number and Rho for wing example.

Used clusters	INRIA-pf	INRIA-nina	IUSTI	CEMEF
Number of processors	8	8	8	8
Communication time(s)	94	21.9	63.8	90.6
Total simulation time(s)	380.4	96.7	116.9	245.8
Used clusters	INRIA-CEMEF	INRIA-IUSTI	CEMEF-IUSTI	
Number of processors	4-4	4-4	4-4	
Communication time(s)	1296.8	1337.3	1542.5	
Total simulation time(s)	1570.6	1438.4	1990	

Table 5: 22.000 vetices example

Acknowledgements This results were provided by Steve Wornom.

STOKES The Stokes code simulates stationary flows for Newtonian fluids in a given meshed cavity. A mixed finite element method of type (P1+/P1) is used (i.e. continuous interpolations for pressure and velocity at the nodes). There are 4 unknowns (Vx, Vy, Vz and P) for each node in a 3D case, and 3 unknowns (Vx, Vy and P) in a 2D case. Once assembled, the global matrix is pre-conditioned by PETSc using ILU(0). Then, PETSc solves the linear system using a conjugate residual method. The meshed cavity is partitioned, then each processor computes the flow inside its own part of the cavity. Communications between processors mostly happens at the interfaces of the partition (i.e. at the nodes that belong to several processors).

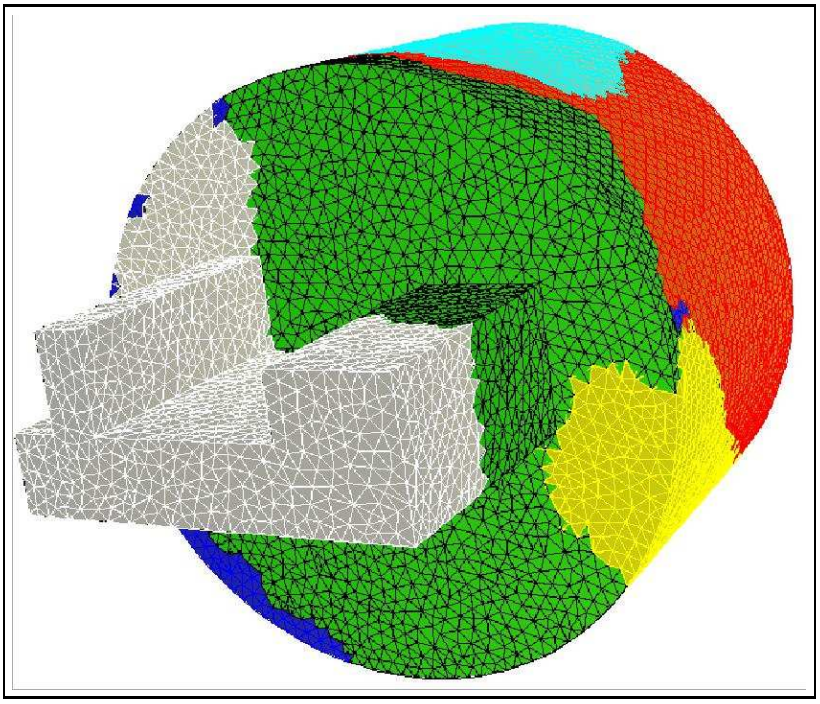

Figure 9: Extrusion

This example is is a 3D extrusion of a profile. The Figure 9 is the result of extrusion on th 65.000 vertice mesh. The table 6 shows the results for 65.000 vertices mesh. The second one 7 shows results for 500.000 vertices mesh. Poor performances of inter-cluster communications are visible in these tables and make necessary to use more larger tests.

Used clusters	INRIA	IUSTI	CEMEF	INRIA-CEMEF-IUSTI	
Number of processors	3	3	3	3 (1-1-1)	9 (3-3-3)
Number of iterations	653	650	653	651	657
Total simulation time(s)	87.56	173.66	455.57	649.52	495.37

Table 6: 65.000 vertices example

Used clusters	INRIA	IUSTI	CEMEF	INRIA-CEMEF-IUSTI
Number of processors	6	6	6	6 (2-2-2)
Number of iterations	1075	1069	1069	651
Total simulation time(s)	692.78	1903.78	1357.1	2995.01

Table 7: 500.000 vertices example

Acknowledgements This results were provided by Olivier Basset.

AEDIPH AEDIPH 3D solves interface problems between two immiscible fluids as well as some two-phase flow problems. The two fluid model is the one described in ref : Guillard-Murrone. The numerical method of the AEDIPH code is an extension of the one used in AERO3D. This code is still in an early stage of developpement and up to day has been mainly used for the computation of some model interface problems.

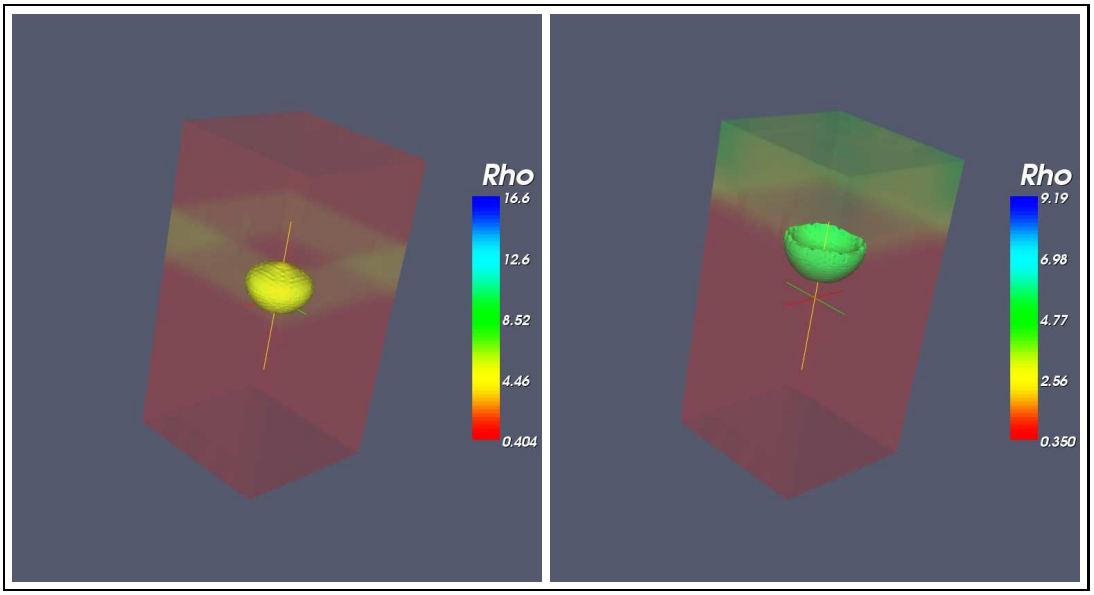

Figure 10: Shock passing 3D bubble, time step 360 and 720

Used clusters	INRIA-pf	INRIA-nina	IUSTI	CEMEF
Number of processors	8	8	8	8
Total simulation time(s)	564.7	331.4	341.2	759.6
Used clusters	INRIA-CEMEF	INRIA-IUSTI	CEMEF-IUSTI	
Number of processors	4-4	4-4	4-4	
Total simulation time(s)	563.4	629.6	828.5	

Table 8: 262.000 vertices example

Used clusters	INRIA-pf	INRIA-nina	IUSTI	CEMEF
Number of processors	16	16	16	16
Total simulation time(s)	760.1	547.3	450.0	1039.8
Used clusters	INRIA-CEMEF	INRIA-IUSTI	CEMEF-IUSTI	
Number of processors	8-8	8-8	8-8	
Total simulation time(s)	1313.5	1204.0	2079.7	

Table 9: 568.000 vertices example

Acknowledgements This results were provided by Steve Wornom.

6 Conclusions and future works

At the sight of our measures of performances, load-balancing and, more generally fault tolerant properties appear essential and should be the main improvements in the future. We plan to carry out some tests with MPICH-V [BBC+02], a fault tolerant layer for MPICH to add this functionnality in our grid. Another effort will be focused on improving the network performances between sites. Some current equipements such as routers and switches have poor performances (10Mbit/s) and we should gain in speed communications by replacing them with 100Mbit/s ones. Then, integration of new clusters (at CEMEF and INRIA, new nodes will be added) in this grid and running of progressively bigger computations are planned.
This document can be seen as a starting point for grid computing on the cluster of INRIA-Sophia and, probably, will be completed later by other technical or research reports from this project or others. We hope that the set of found solutions will evolve with time and, in the future, we will use a flexible and totally integrated middleware for grid computing.
On this topic, we hope to see a lot of improvements in grid computing from the "Grid5000" project whom INRIA-Sophia is a partner. We hope this project, which have a similar approach, could be the starting point of a French alternative middleware for grid computing. Moreover, it could be an appropriate testbed for libraries as MADELEINE that could use specialized links as INRIA-Grenoble/INRIA-Sophia.

References

[AM] Olivier Aumage and Guillaume Mercier. Mpich/madiii: a cluster of clusters enabled mpi implementation.

[BBC+02] George Bosilca, Aurelien Bouteiller, Franck Cappello, Samir Djilali, Gilles Fedak, Cecile Germain, Thomas Herault, Pierre Lemarinier, Oleg Lodygensky, Frederic Magniette, Vincent Neri, and Anton Selikhov. Mpich-v: toward a scalable fault tolerant mpi for volatile nodes. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–18. IEEE Computer Society Press, 2002.

[CKKG99] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew S. Grimshaw. The Legion resource management system. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 162–178. Springer Verlag, 1999.

[Con] VPN Consortium. Vpn technologies: Definitions and requirements.

[ES01] Dietmar W. Erwin and David F. Snelling. UNICORE: A Grid computing environment. *Lecture Notes in Computer Science*, 2150:825–??, 2001.

[FK97] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[GL96] William D. Gropp and Ewing Lusk. *User's Guide for* `mpich`*, a Portable Implementation of MPI.* Mathematics and Computer Science Division, Argonne National Laboratory, 1996. ANL-96/6.

[gri03] http://wwws.sun.com/software/gridware/. Website, 2003.

[Kar] Nicholas T. Karohis. Mpich-g2: A grid-enabled implementation of the message passing interface.

[Lan03] Rodolphe Lanrivain. Partitionnement de maillages sur une grille de calcul. Rapport de stage DEA, Université de Bordeaux, juin 2003.

[lsf01] http://www.platform.com/products/LSFfamily/. Website, May 2001.

[ope03] http://www.openpbs.org/. Website, May 2003.

[pac03] http://www.hlrs.de/organization/pds/projects/pacx-mpi/. Website, 2003.

[TC88] International International Telephone and Telegraph Consultative Committee. Information technology - open systems interconnection - the directory: Authentication framework (ccitt recommendation x.509), 1988.

[Tit03] Olaf Titz. http://sites.inka.de/sites/bigred/devel/cipe.html. Website, 2003.

[TTL02] Douglas Thain, Todd Tannenbaum, and Miron Livny. Condor and the grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

[WD96] D. W. Walker and J. J. Dongarra. MPI: a standard Message Passing Interface. *Supercomputer*, 12(1):56–68, 1996.

[Ylo96] T. Ylonen. Ssh—secure login connections over the internet. *In Proceedings of the 6th USENIX Security Symposium*, pages 37–42, 1996.

Contents

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803